UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

SPACECRAFT CONSTELLATION PLANNING FACILITY
PLANNING REQUEST HANDLER

PROJECTO REALIZADO NA

CRITICAL SOFTWARE, SA

POR

CÁTIA REGINA CASTRO TORRES

MESTRADO EM ENGENHARIA INFORMÁTICA

2007

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

SPACECRAFT CONSTELLATION PLANNING FACILITY
PLANNING REQUEST HANDLER

projecto realizado na

Critical Software, SA

por

Cátia Regina Castro Torres

Projecto orientado pelo Prf. Dr João Pedro Neto
e co-orientado por Flávio Moreira

Mestrado em Engenharia Informática

2007

Declaração

Cátia Regina Castro Torres, aluno nº 30398 da Faculdade de Ciências da Universidade de Lisboa, declara ceder os seus direitos de cópia sobre o seu Relatório de Projecto em Engenharia Informática, intitulado "Spacecraft Constellation Planning Facility – Planning Request Handler" , realizado no ano lectivo de 2006/2007 à Faculdade de Ciências da Universidade de Lisboa para o efeito de arquivo e consulta nas suas bibliotecas e publicação do mesmo em formato electrónico na Internet.

Lisboa, 1 de Junho de 2007

_____

Flávio Moreira, supervisor do projecto de Cátia Regina Castro Torres, aluno da Faculdade de Ciências da Universidade de Lisboa, declara concordar com a divulgação do Relatório do Projecto em Engenharia Informática, intitulado "Spacecraft Constellation Planning Facility – Planning Request Handler".

Lisboa, 1 de Junho de 2007

_____

Agradecimentos

Numa etapa importante a nível académico e pessoal, foram várias as pessoas que me apoiaram e que contribuíram para que esta fase fosse superada com sucesso. Deixo aqui, então, um agradecimento especial a todas elas.

A todos os colegas da Critical Software, SA agradeço o facto de me proporcionarem um agradável e estimulante ambiente de trabalho, mostrando-se sempre disponíveis para me ajudarem. Um agradecimento especial à equipa de projecto em que participei, nomeadamente ao Flávio Moreira, Tiago Franco e Henrique Oliveira, que me ajudaram a desenvolver o meu trabalho oferecendo sugestões e críticas construtivas para o melhoramento do meu trabalho conduzindo ao meu crescimento profissional.

Ao Prof. João Neto por se ter mostrado sempre disponível.

Aos meus pais, irmã e avó que me apoiaram e acreditarem que eu seria capaz de chegar ao fim, e acima de tudo por suportarem o meu mau humor em momentos de maior stress.

Aos meus amigos por “estarem sempre lá”, por me ajudarem a desanuviar nas alturas de maior pressão e por me darem sempre força para continuar.

E finalmente, ao meu namorado, João, também colega de faculdade e trabalho, pela compreensão, apoio, ajuda, por me ter dado ânimo nos momentos menos felizes; essencialmente por ter estado sempre presente.

Resumo

No futuro que se aproxima, será cada vez mais necessário determinar a posição precisa de alguém, no espaço e no tempo, de uma forma fidedigna. Dentro de poucos anos esta necessidade vai ser coberta pelo sistema de navegação de rádio satélite Galileo, uma iniciativa lançada pela União Europeia (UE) e a Agência Espacial Europeia (ESA). Este sistema assegurará a sua complementaridade com os actuais sistemas de posicionamento existentes, o GPS, americano, e o GLONASS, russo.

A missão Galileo é constituída por uma constelação de 30 satélites que serão coordenados através do Spacecraft Constellation Planning Facility (SCPF). Integrado no SCPF, existe um componente, o Planning Request Handler (PRQ), responsável por receber pedidos de controlo dos satélites por parte de outros sistemas. O PRQ tem como principais competências a validação dos pedidos recebidos e a sua inserção num repositório de dados. O PRQ é constituído por um servidor e uma pequena aplicação gráfica que permitirá a submissão de pedidos no sistema por um operador.

Este documento aborda todo o trabalho realizado no desenvolvimento do componente Planning Request Handler durante o estágio: desenho, implementação e validação.

Este documento pretende, além de demonstrar o trabalho realizado, transmitir uma apreciação da minha experiência na entrada no mundo profissional e do estágio.

PALAVRAS-CHAVE: Galileo, Planning Request Handler, SCPF, Pedidos, Validação.

Abstract

In the near future, the need to determinate the exact position of someone, in time and space, will become more and more important. In a few years, this need will be fulfilled by the radio-satellite navigation system Galileo, a project launched by the European Union (UE) and the European Space Agency (ESA). This system will ensure its complementarity with the already existent positioning systems, the american GPS and the russian GLONASS.

The Galileo mission is constituted by a constellation of 30 satellites that will be coordinated by the Spacecraft Constellation Planning facility (SCPF). The Planning Request Handler (PRQ) is one of the SCPF components. Its responsibility is to receive requests to control satellites from other systems. The main responsibility of the PRQ is to validate the received requests and then to store them in a data store. The PRQ is constituted by a server and a small graphic application that will allow the submission of planning requests by an operator.

This document describes the whole work realised in the Planning Request Handler development during the internship: design, implementation, verification and validation.

This report intends, beyond showcasing the accomplished work, to assess my entrance in the professional world and the internship.

KEYWORDS: Galileo, Planning Request Handler, SCPF, Requests, Validation.

Índice

Índice de Figuras

Índice de Tabelas

Capítulo 1

Introdução

No futuro que se aproxima, será cada vez mais necessário determinar a posição precisa de algo ou alguém, no espaço e no tempo, de uma forma fidedigna. Dentro de poucos anos esta necessidade vai ser coberta, de forma independente na Europa, pelo sistema de navegação de rádio satélite Galileo, uma iniciativa lançada pelo União Europeia (UE) e a Agência Espacial Europeia (ESA). Desenhado para garantir a sua interoperabilidade com os dois sistemas semelhantes já existentes, o americano Global Positioning System (GPS) e o russo Global Orbiting Navigation Satellite System (GLONASS), o Galileo irá permitir aos utilizadores localizarem um ponto geográfico a partir de qualquer combinação de satélites através de um único receptor. Tanto o GLONASS como o GPS são geridos e mantidos pelos departamentos de defesa dos respectivos países. O Galileo irá ser operado por civis.

A curto prazo, os equipamentos de determinação de posição por satélite tornar-se-ão para todos nós tão indispensáveis como, hoje em dia, é o relógio. Dentro em breve, cada telefone móvel terá, assim, a capacidade para receber os sinais emitidos pelos satélites e oferecerá a possibilidade de conhecer a qualquer momento e em qualquer ponto do globo a posição das pessoas, dos veículos, dos navios, dos aviões, das mercadorias, dos animais. Esta tecnologia melhorará significativamente os sistemas de orientação, a prevenção dos acidentes, a eficácia da protecção civil (chamadas de emergência ou pedidos de socorro) e a protecção do ambiente. Particulares, empresas, administrações, todos poderão beneficiar do sistema, na estrada, na via férrea, no ar e no mar: os praticantes de pedestrianismo saberão encontrar o seu caminho, os turistas o seu museu ou restaurante, os taxistas a morada certa.

1.1. Âmbito do projecto

O estágio decorreu no âmbito da disciplina de Projecto em Engenharia Informática do Mestrado em Engenharia Informática (MEI) da Faculdade de Ciências da Universidade de Lisboa.

O projecto do estágio insere-se na missão Galileo, uma inciativa lançada pela União Europeia(UE) e a Agência Espacial Europeia (ESA), cujo objectivo consiste em oferecer um sistema de posicionamento geográfico com mais funcionalidades, mais fiável e independente, mas complementar, dos já existentes.

O trabalho consiste no desenvolvimento de um componente (Planning Request Handler), pertencente a um sistema de planeamento de uma constelação de satélites, que é responsável pela recepção de pedidos de outros sistemas do Ground Control Center do Galileo, assim como do próprio sistema em que está inserido. Os pedidos recebidos são alvos de um processo de validação, após o qual são geradas respostas, a ser enviadas aos seus emissores, com dados sobre a validação. Tanto os dados recebidos como os gerados são guardados num repositório de dados.

1.2. Objectivos

O objectivo do projecto é o desenvolvimento do componente Planning Request Handler (PRQ), que está inserido no Spacecraft Constellation Planning Facility (SCPF), incluindo as fases de desenho, especificação de planos de testes unitários e de aceitação, implementação e verificação e validação do componente, assim como geração da documentação respeitante ao seu desenvolvimento.

1.3. Contexto Institucional

A disciplina de Projecto em Engenharia Informática, na qual se insere este estágio, é realizada, tipicamente, numa instituição de acolhimento, sendo a referente a este projecto, a Critical Software S.A.

A Critical Software S.A. é uma empresa internacional, fundada em Portugal em 1998, estando sedeada em Coimbra, com escritórios em Lisboa, Porto, San Jose (EUA) e Londres (Reino Unido). A Empresa desenvolve soluções, serviços e tecnologias inovadoras e fiáveis, para sistemas críticos de negócios dos seus parceiros e clientes empresariais. Os seus clientes estão situados em todos os continentes e operam nos sectores da Aeronáutica, Banca, Defesa, Espaço, Indústria, sector Público e Telecomunicações, destacando-se a Agência Espacial Europeia, a NASA, a Marinha Portuguesa, entre outros.

A empresa é certificada ISO 9001:2000 Tick-IT (British Standard Institute), seguindo processos de qualidade rigorosos. Os pontos fortes da empresa residem na confiabilidade e qualidade de software, assim como na inovação tecnológica.

1.4. Acrónimos e definições

Acrónimo	Definição
CSW	Critical Software, S.A.
DOM	Document Object Model
EA	Enterprise Architect
FDF	Flight Dynamics Facility
GCS	Ground Control Segment
GMS	Ground Mission System
GUI	Graphical User Interface
GSWS	Galileo Software Standard
IDE	Integrated Development Environment (applications like Eclipse, Netbeans)
JVM	Java Virtual Machine
MTP	Mid-Term Plan
PDS	Planning Data Store
PRH	Planning Request Server, a PRQ subcomponent
PRQ	Planning Request Handler, an SCPF software component
PRU	Planning Request Handler User Interface
SAX	Simple API for XML
SCPF	Spacecraft Constellation Planning Facility

Acrónimo	Definição
SWT	Standard Widget Toolkit
TT&C	Telemetry, Tracking and Command
UML	Unified Modelling Language
XML	Extensible Mark-up Language

Tabela 1: Acrónimos e definições

1.5. Organização do documento

O Capítulo 1 contém uma breve introdução ao projecto, o âmbito em que foi realizado, objectivos, tecnologias e ferramentas utilizadas, e o contexto institucional em que se enquadra.

O Capítulo 2 apresenta o contexto em que se desenvolveu o estágio e os métodos de trabalho utilizados.

O Capítulo 3 apresenta as fases de desenvolvimento do Planning Request Handler.

O Capítulo 4 apresenta uma apreciação e conclusão acerca do trabalho realizado e do estágio.

Capítulo 2

Contexto do projecto e métodos utilizados

Neste capítulo é apresentado o contexto subjacente do projecto, nomeadamente a contextualização no sistema Galileo, e os métodos de trabalho seguidos durante o seu desenvolvimento.

2.1. Sistema Galileo

O Galileo é um sistema de posicionamento geográfico que tem os seguintes objectivos:

- Oferecer uma maior precisão do que a actualmente existente;
- Fornecer melhor cobertura dos sinais emitidos pelos satélites a altas latitudes, da qual as regiões nórdicas, como a Escandinávia, irão beneficiar;
- Disponibilizar um sistema de posicionamento do qual as nações europeias podem usufruir mesmo em tempos de guerra e divergências políticas.

Além destes aspectos, o Galileo irá oferecer uma maior segurança aos seus utilizadores, já que irá notificá-los, em poucos segundos, da falha de um satélite.

O sistema Galileo irá englobar vários componentes: global, regional e locais.

O componente global é o cerne do sistema; abrange os satélites e o Ground Control Segment (GCS).

O componente regional do Galileo pode abranger External Region Integrity Systems (ERIS), implementados e operados por organizações, países, ou grupos de países, fora da Europa para obter serviços de integridade independentes do sistema Galileo.

Os componentes locais podem ser distribuídos para melhorar o desempenho do Galileo a nível local. Estes podem contribuir para a melhoria da recepção de sinal de navegação em áreas onde os sinais dos satélites não são recebidos.

2.1.1 Componente Global do Galileo

O componente global do Galileo irá englobar a constelação de satélites do sistema. Cada um dos satélites será responsável pela propagação de sinais de tempo de navegação em conjunto com sinais de dados de navegação que irão conter, além dos dados essenciais para a navegação, sinais de integridade.

O segmento espacial irá ser complementado com o Ground Control Segment, que irá compreender dois centros de controlo e uma rede global de estações para recepção e transmissão de dados.

2.1.1.1. Ground Mission Segment

O Ground Mission Segment (GMS) irá usufruir de uma rede global constituída por trinta estações de sensores (Galileo Sensor Stations - GSS) para monitorizar os sinais de navegação de todos os satélites, através de uma rede de comunicação abrangente, utilizando satélites comerciais, assim como ligações por cabo. Cada uma destas será duplicada para garantir redundância.

2.1.1.2. Ground Control Segment

O Galileo Ground Segment será constituído por dois centros de controlo. Cada um destes centros irá gerir actividades de controlo, suportadas por um Ground Control Segment (GCS) dedicado, e actividades de missão, suportadas por um Ground Mission Segment (GMS). O GCS é responsável pelas operações de manutenção, monitorização e controlo da constelação de satélites, enquanto o GMS gere o controlo do sistema de navegação. O GCS irá usufruir de uma rede global de cinco estações TT&C para a comunicação com cada satélite.

A Figura 1 apresenta a interacção entre o GCS e outras entidades.

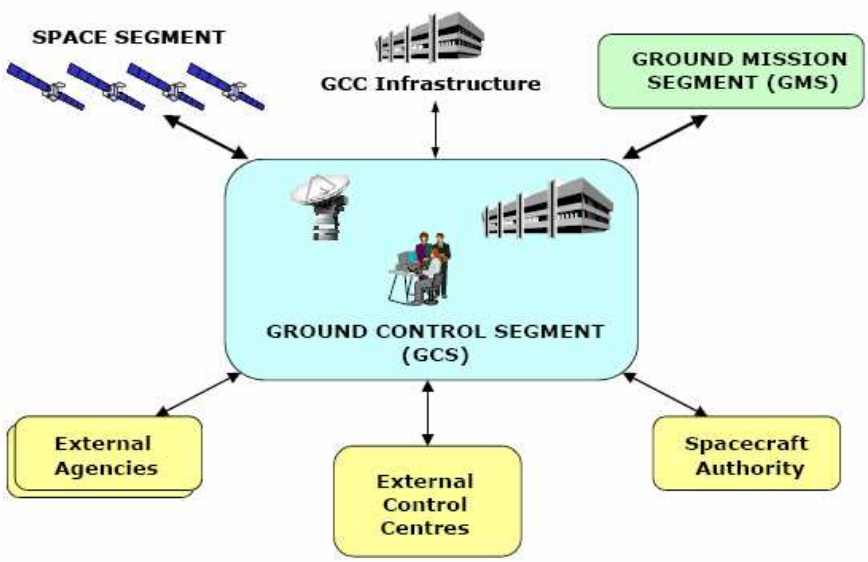

Figura 1: Ground Control Segment

O GCS interage com o GMS para a coordenação de operações e administração dos recursos terrestres como estações e redes.

As outras entidades externas (External Agencies, External Control Centres e Spacecraft Authority) fornecem serviços ao GCS, mas não estão em contacto permanente com este.

2.1.1.2.1 Spacecraft Constellation Planning Facility

O *Spacecraft Constellation Planning Facility* (SCPF) é responsável pelas operações de planeamento do GCS.

O processo nominal de planeamento é iniciado com o processamento do *Mid-Term Plan* (um conjunto de *planning requests* e informação relativa a disponibilidade das estações TT&C) e todos os pedidos referentes a operações sobre a constelação de satélites (recebidos do GMS, FDF (*Flight Dynamics Facility*) e operador do SCPF). Após esta fase, procede-se à geração do plano de contacto, que servirá como input ao componente responsável pelo planeamento de tarefas e actividades . O resultado final deste processo tem a forma de *Short-Term Plan*, um plano de operações a realizar a curto prazo, que é depois enviado ao *Spacecraft Constellation Control Facility* para execução.

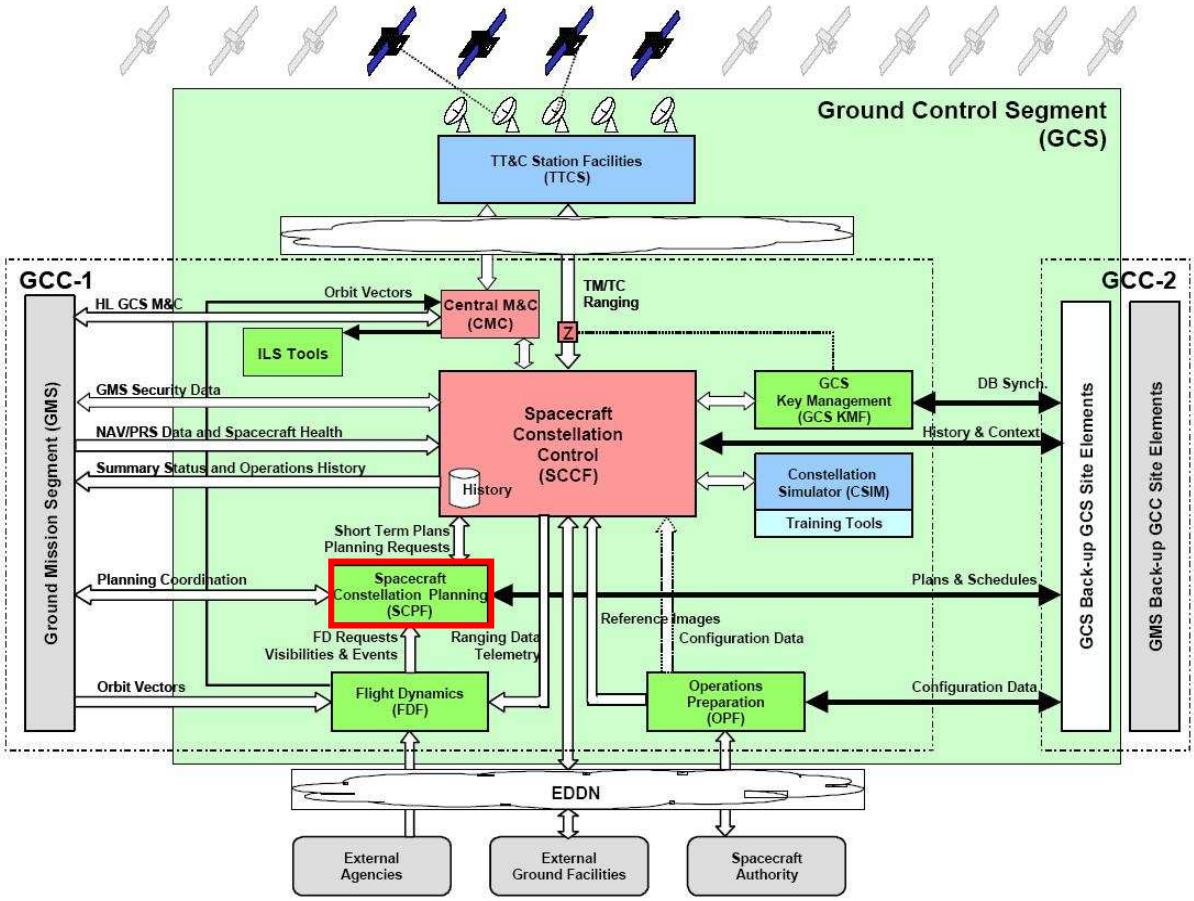

Figura 2: Contextualização do SCPF no GCS

A Figura 2 dá-nos uma ideia do contexto em que o SCPF se enquadra no Ground Control Segment (GCS). Além do SCPF os outros elementos da figura são:

- *Spacecraft Constellation Control Facility* (SCCF) – Efectua operações de monitorização e controlo dos satélites.
- *Flight Dynamics Facility* (FDF) – Suporta o cálculo de órbitas, planeamento de manobras e monitorização do comportamento individual dos satélites e da própria constelação.
- *Operations Preparation Facility* (OPF) – Responsável pela preparação e configuração de controlo de todos os procedimentos operacionais, incluindo aqueles que virão a ser realizados de modo automático.
- *Key Management Facility* (KMF) – Este elemento está relacionado com os aspectos de segurança das missões.
- *Central Monitoring and Control Facility* (CMCF) – Suporta a monitorização e controlo de todas as estações terrestres do GCS.
- *Simulator, Training and Integrated Logistic Support Tools.*

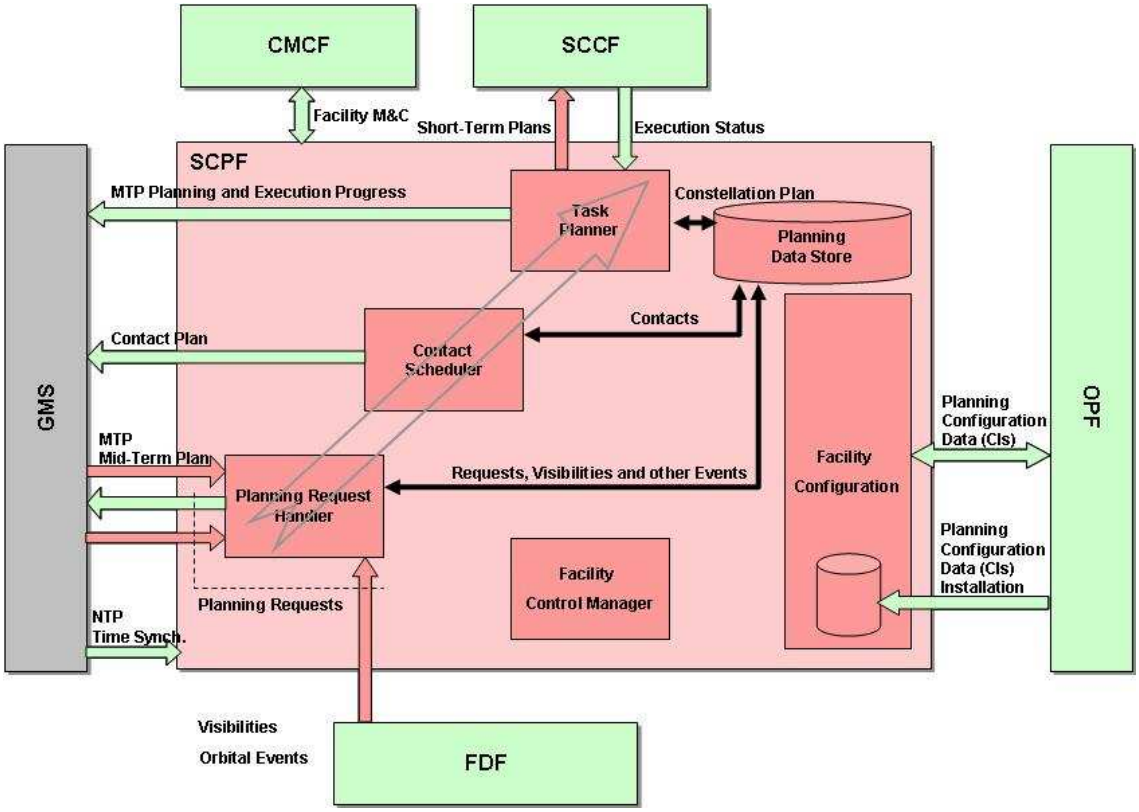

Figura 3: Arquitectura do SCPF

O SCPF (Figura 3) é responsável pelo planeamento a curto termo das operações da constelação de satélites do Galileo. Esta actividade requer coordenação com o GMS, pois este é responsável pela gestão da provisão de serviços do Galileo e da geração do *Mission Plan*, que consiste no *input* inicial para a função de planeamento do GCS.

As suas principais funções são:

- *Gestão de pedidos*: o *Planning Request Handler* (PRQ) recebe pedidos de planeamento por parte do FDF (componente responsável pela determinação de toda a dinâmica da constelação de satélites) e do operador, encaminhando os que têm *mission impact* (operações que possam comprometer os serviços da missão) para o GMS. Além dos referidos pedidos, o PRQ também recebe o *Mid-Term Plan* enviado pelo GMS. Todos os pedidos recebidos pelo PRQ, assim como os que fazem parte do Mid-Term Plan, são validados e armazenados. O resultado da validação é enviado aos emissores.
- *Escalonamento de contactos*: o *Contact Scheduler* aceita dados sobre a disponibilidade das estações TT&C (i.e., informação relativa aos intervalos de tempo em que estas podem ser contactadas), visibilidades da constelação de satélites das estações TT&C (intervalos em que é possível as estações TT&C contactarem a constelação de satélites) e pedidos de planeamento que requeiram contacto. Com a utilização destes dados, entre outros, cria um plano de contacto dentro dos limites dos recursos indicados.
- *Planeamento de tarefas*: o *Task Planner* aceita *orbit events* (dados sobre a dinâmica das órbitas dos satélites) do Flight Dynamics Facility (FDF), um plano de contacto do Contact Scheduler e um Mid-Term Plan do GMS através do PRQ. Os eventos e os contactos são usados para activar tarefas e actividades

num plano. Após validação do plano pelo operador é criado o Short-Term Plan (um plano de contacto que cobre um determinado número de dias em avanço).

- *Editor da base de dados de planeamento*: as funções acima descritas são realizadas com base em definições configuráveis armazenadas em bases de dados. Estas definições são guardadas na *Facility Configuration*, uma base de dados relacional que disponibiliza também uma aplicação para edição.

Os ficheiros recebidos, gerados e enviados pelo SCPF são armazenados na *Planning Datastore (PDS)*, um repositório de dados usados para o planeamento.

Planning Request Handler

Os dados recebidos pelo SCPF, indispensáveis à sua função de planeamento, consistem em pedidos para a execução de tarefas e actividades sobre a constelação de satélites e estações do Galileo, e eventos referentes à mesma constelação.

Todos os pedidos são recebidos pelo *Planning Request Handler*, que é responsável pela sua validação e armazenamento num repositório de dados local ao SCPF. Após a validação é enviada uma resposta com o resultado da mesma (aceite ou rejeitado) ao emissor do pedido. Os pedidos aceites são posteriormente planeados constituindo parte de um *Short-Term Plan* (STP). O *Planning Request Handler* é composto por um servidor responsável por todo o processo de recepção e validação de pedidos, assim como a geração de respostas, (também procede ao armazenamento de ambos); e por uma GUI que permitirá a um operador submeter pedidos e editar outros pedidos já validados.

Na análise da criticalidade do componente foi detectada uma barreira de segurança no servidor que condiziria à divisão do servidor em dois subcomponentes. Esta barreira de segurança foi implementada ao nível da validação dos ficheiros recebidos resultando na necessidade de garantir de todos os ficheiros geridos pelo PRQ. Apenas passam para a fase seguinte de planeamento os ficheiros que sejam válidos segundo as suas definições.

2.2. Métodos de trabalho

No desenvolvimento do trabalho foram adoptadas estratégias a nível de processos de trabalho com o objectivo de realizar um trabalho com qualidade.

Os processos de qualidade têm como objectivos os seguintes pontos:

- Conhecer os requisitos do cliente;
- Entregar um trabalho de qualidade assegurada no tempo planeado, dentro do orçamento estipulado;
- Implementar um sistema de gestão qualidade efectivo;
- Promover o melhoramento contínuo.

2.2.2 Reuniões de projecto

Semanalmente foram realizadas reuniões de projecto nas quais foi discutido o estado de progresso do projecto. Para cada reunião existe uma agenda com os pontos a discutir que geralmente consistem nos seguintes: estado da execução das acções estipuladas na reunião anterior, revisão do plano de projecto e estado dos *work-packages* e definição de acções a serem executadas por cada elemento da equipa.

2.2.2.1. Análise de riscos

Pontualmente nas reuniões de projecto foi efectuada uma análise dos riscos inerentes ao projecto, isto é, a identificação e análise das ameaças e oportunidades. Apesar desta actividade se centrar sobretudo nos riscos que podem colocar em causa o desenvolvimento do projecto dentro do plano estabelecido, assim como a qualidade do mesmo, também deverá incluir a promoção dos eventos positivos.

A análise de riscos consistiu na identificação de novos riscos e na análise do estado dos riscos já estabelecidos. Para cada risco existe um conjunto de atributos associados:

- Identificação do risco: palavra que identifica unicamente cada risco
- Descrição do risco
- Associação de acções que permitam corrigir, mitigar e controlar o risco, ou promovê-lo caso seja um risco positivo.
- Estado da acção associada: aberta ou fechada
- Origem do risco: externa ou interna
- Probabilidade de ocorrência do risco
- Impacto: grau de severidade do risco; medido pela gravidade dos problemas que surgiriam no projecto desencadeados pela ocorrência do risco
- Estado: estado actual do risco. Os estados possíveis são: identificado, monotorizado, e fechado
- Severidade: grau de importância do risco, calculado com base na sua probabilidade e impacto

2.2.3 Comunicação com o cliente

A comunicação com o cliente é um factor bastante importante durante o desenvolvimento de um projecto. Esta efectuou-se na maioria das vezes por correio electrónico pois foi o meio mais fácil e rápido de utilizar. A linguagem de comunicação foi o inglês, visto o cliente ser estrangeiro, nomeadamente britânico.

Em Novembro teve lugar uma reunião com o cliente nas instalações da Critical Software, S.A., permitindo um contacto mais directo com o cliente. Nesta reunião foi apresentado o trabalho até então desenvolvido, discussão do mesmo e de alguns pontos relativos a nova documentação.

Outras reuniões realizadas no exterior, com mais entidades envolvidas, tiveram a participação do gestor de projecto. As decisões tomadas foram depois transmitidas à equipa nas reuniões de projecto.

2.2.4 Sistema de gestão de qualidade

O desenvolvimento de software segue um conjunto de processos definidos no Sistema de Gestão da Qualidade. Esses processos estão identificados no Manual de Qualidade e são necessariamente suportados por procedimentos, guias, *checklists*, templates entre outros documentos.

O Sistema de Gestão da Qualidade foi desenhado para permitir que os projectos e outras actividades atinjam os seus objectivos de foma eficiente, pelo uso de processos e práticas definidos, garantindo a sua qualidade. Para todos os processos são produzidos documentos que permitam documentar a realização dos processos, servindo assim de

prova e de objecto de consulta. Alguns dos documentos produzidos para vários processos:

- Auditorias: plano, relatório e acções correctivas e preventivas
- Projectos: plano do projecto, plano de manutenção, entregas do produto, revisões
- Clientes: proposta, contractos, minutas de reuniões externas
- Gestão: política de qualidade e objectivos, standards a serem seguidos no desenvolvimento de software

O SCPF sofreu duas auditorias, uma a nível interno, e outra a nível externo. As auditorias internas são planeadas e realizadas pelo Departamento de Qualidade e envolve além do auditor a equipa do projecto alvo; o seu principal objectivo é certificar que o projecto cumpre os processos e regras de qualidade definidas, resultando um conjunto de acções correctivas e preventivas.

As auditorias externas seguem um processo semelhante às internas e são realizadas por um entidade independente.

Todos os documentos produzidos durante o estágio sofreram revisões por parte dos outros elementos do projecto. A maioria das revisões do projecto consistiram essencialmente por uma primeira fase, na qual os revisores reviram o documento, anotando comentários, críticas e sugestões; segue-se uma segunda fase, na qual os revisores e o autor do documento se reúnem e são discutidos todos os pontos que apresentem incoerências, necessitem melhorias, ou tenham lacunas. O autor do documento reúne todas as decisões tomadas na discussão e implementa-as no documento produzindo uma segunda versão do mesmo. Todas as revisões têm um relatório onde é documentado quais os participante envolvidos, e quais os erros ou melhorias a aplicar.

2.2.4.1. Standards de Software do Galileo

Sendo o SCPF um projecto incluído na missão Galileo tem que seguir um conjunto de processos partilhados por todos os projectos desta missão de modo a uniformizar os produtos desenvolvidos e garantir a sua qualidade.

O *Galileo Software Standard* (GSWS) estabelece os requisitos que devem ser seguidos na gestão, avaliação, desenvolvimento, produção, verificação, operação e manutenção de todos os produtos de software do Galileo.

Estes processos são suportados por templates, guias e checklists que deverão fazer parte do sistema de qualidade do projecto.

As auditorias realizadas centraram-se fundamentalmente na certificação do seguimento dos processos do Galileo.

2.2.5 Sistema de Controlo de Versões (CVS)

Toda a documentação produzida no âmbito do projecto é centralizada num directório sob um sistema de controlo de versões (CVS). O CVS é um sistema que guarda o registo histórico das alterações efectuadas sobre um ou vários ficheiros. Além de permitir a centralização de toda a informação acedida pela equipa do projecto e a possibilidade de aceder a versões anteriores de um mesmo documento, permite que os utilizadores trabalhem sobre uma cópia local de um documento, enviando posteriormente as mudanças para o servidor.

Capítulo 3

Planning Request Handler : trabalho realizado

O trabalho realizado no desenvolvimento do Planning Request Handler envolveu as seguintes fases: desenho, implementação e validação. Durante estas etapas foram seguidas algumas metodologias de trabalho de forma a garantir a qualidade e coerência dos resultados conseguidos, tivessem eles a forma de documentos ou código fonte.

Todas as tecnologias e ferramentas a utilizar já estavam definidas no início do estágio.

3.1. Análise de Requisitos

A análise de requisitos é o estudo das características que o sistema deverá possuir para atender às necessidades e expectativas do cliente. Cada funcionalidade requirida pelo cliente deverá ser analisada para verificar os possíveis impactos no desenvolvimento das demais funcionalidades do sistema. A disponibilização das tecnologias que permitem a implementação das funcionalidades também deve ser verificada.

A identificação dos requisitos já tinha sido efectuada no início do estágio.

Esta fase consistiu inicialmente na leitura de documentação técnica relativa ao SCPF, de modo a conseguir uma maior familiarização com o sistema. Os documentos analisados foram:

- Documento de Especificação do Desenho do SCPF – documento preliminar que especifica o SCPF, constitui o documento de arquitectura do SCPF.
- Documentos de Propostas da Definição de Interfaces – documentos que especificam as interfaces entre o SCPF e os outros componentes do Galileo (FDF, GMS, SCCF) definindo as mensagens trocadas entre si.
- Documento de Especificação do Sistema – identifica os requisitos de cada subcomponente do SCPF. A análise deste documento centrou-se no capítulo referente ao *Planning Request Handler*.

3.2. Desenho detalhado

O desenho detalhado de um sistema de software tem o propósito de incitar à tomada do maior número possível de decisões, no desenho do sistema, a partir dos requisitos do sistema definidos, de modo a economizar esforço na fase de implementação; permite fornecer uma base de desenho do sistema que poderá passar por várias revisões; e munir os programadores de documentação (direcções) para a implementação, mais propriamente, o modo como as estruturas de dados e de controlo estarão organizadas.

A fase de desenho é iterativa. O seu principal objectivo é criar um desenho que seja simples, compreensível, de fácil construção e facilmente testável. A linguagem universalmente utilizada para a documentação do desenho é a *Unified Modeling Language* (UML).

Para além dos requisitos, também serviram de input os documentos de definição das interfaces e o documento da especificação do desenho do SCPF (elaborados pelo cliente).

O resultado desta fase constitui o documento de desenho detalhado. O modelo UML encontra-se num documento à parte. O cliente é o responsável pela elaboração de um único documento que integre todo o desenho detalhado relativo ao sistema SCPF.

3.2.1 Desenvolvimento

A primeira actividade desta fase foi a identificação das estruturas e funcionalidades do *Planning Request Handler* procedendo ao seu agrupamento e definição de componentes. Para isso recorreu-se à linguagem UML.

O UML disponibiliza um conjunto de diagramas para repesentação do modelo do sistema, que se enquadram em dois grandes grupos, *Diagramas Estruturais* e *Diagramas Comportamentais.*

- Diagramas Estruturais – dão uma visão estática da estrutura de suporte ao sistema, sobre o qual ele será construído. Para a modelação da estrutura do PRQ recorreu-se aos diagramas de classe e de componentes.
    - Diagrama de classes – descreve a estrutura do sistema, apresentando as classes e interfaces do sistema, os seus atributos e as relações entre si. Uma classe identifica entidades importantes do domínio a que se refere.
    - Diagrama de componentes – ilustra como as classes se deverão encontrar organizadas através da noção de componentes, ou seja, como um sistema se divide em unidades físicas, demonstrando as dependências entre si.
- Diagramas Comportamentais - focam-se no fluxo do sistema, considerando os seus aspectos dinâmicos. Para a modelização do comportamento do PRQ recorreu-se aos diagramas se sequência e de estados.
    - Diagrama de sequência – descreve a maneira como os grupos de objectos colaboram ao longo do tempo num determinado comportamento, exibindo os objectos e mensagens trocadas.
    - Diagrama de estados – ilustra o modo como um elemento transita entre estados, isto é, como muda de condição durante o seu ciclo de vida. Cada estado representa uma condição na qual o elemento pode residir enquanto satisfaz alguma condição, executa uma actividade ou espera por um evento.

Existiam logo à partida dois componentes definidos: o servidor e o cliente. Foi a partir desta base que se procurou encontrar e definir as funcionalidades de cada um, assim como a identificação das funcionalidades comuns.

As classes que definiam funcionalidades partilhadas pelo cliente e pelo servidor foram incluídas num componente à parte acedido pelas duas partes, chamado *Utilities*. Este componente fornece serviços comuns utilizados tanto pelo servidor como pelo cliente.

A Figura 4 apresenta um exemplo de diagrama UML que corresponde a um diagrama de componentes do *Planning Request Handler*.

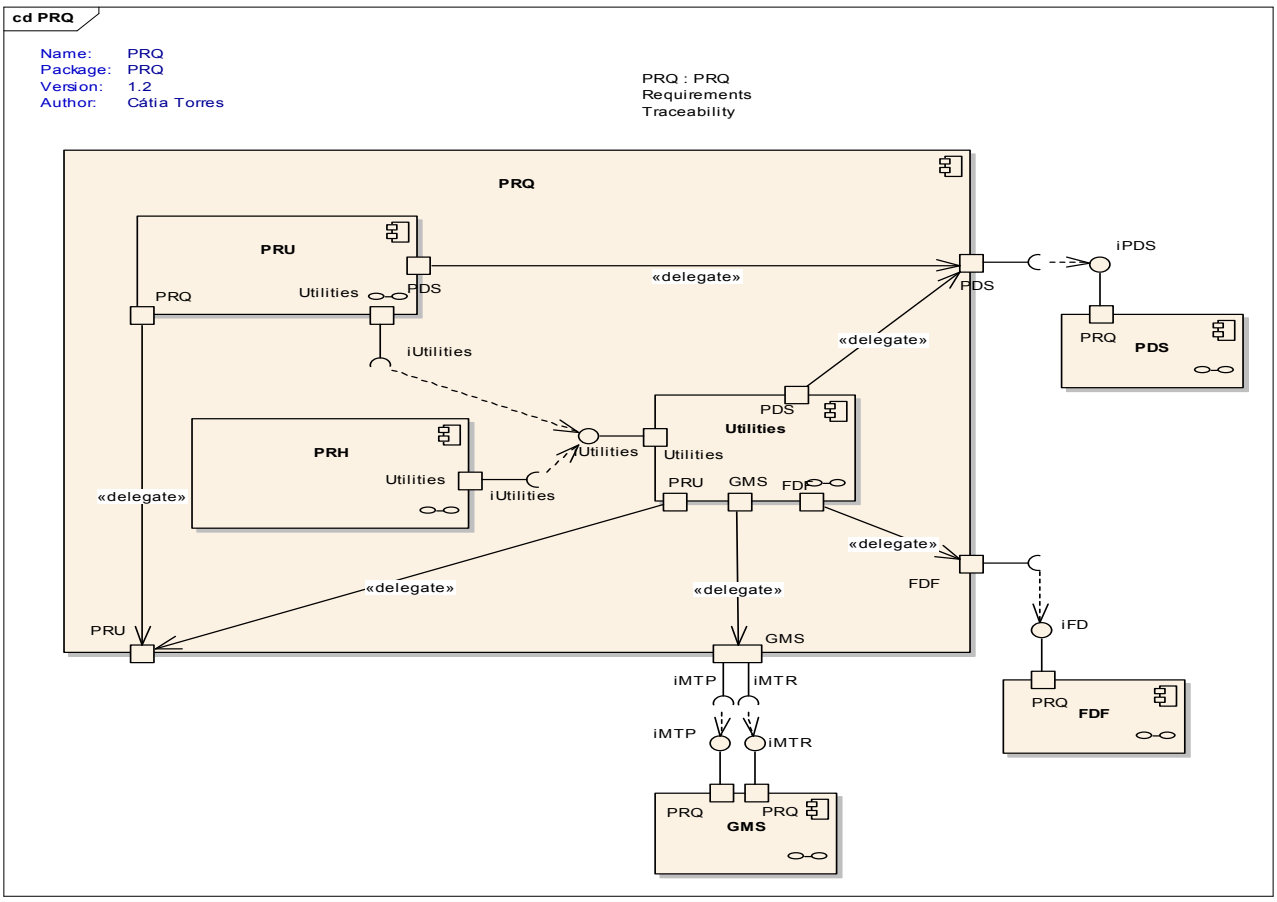

Figura 4: Componentes do PRQ

- *PRH – Planning Request Handler Server*, o servidor do PRQ, responsável pela recepção de pedidos, de produtos orbitais e do *Mid-Term Plan (MTP)*, extrai os pedidos contidos nestes assim como as visibilidades das estações. Valida todos os pedidos e gera uma resposta com o resultado que é enviada ao emissor do pedido. Todos os ficheiros recebidos e gerados são armazenados num repositório de dados (*Planning Datastore - PDS*).
- *PRU – Planning Request Handler User Interface*, o cliente do PRQ. Permite a criação de novos pedidos e a edição dos já existentes. Os novos pedidos são submetidos ao PRH para validação. A resposta resultante da validação dos pedidos submetidos através do PRU é exibida no seu ecrã.
- *Utilities* – componente do PRQ que engloba serviços acedidos tanto pelo PRH como pelo PRU, como por exemplo o *logging*.
- *iMTP* – interface pela qual o PRQ recebe o *Mid-Term Plan*.
- *iMTR* – interface utilizada pelo PRQ para reencaminhamento dos pedidos identificados como tendo impacto na missão e para o envio de respostas para o GMS.
- *iFD* – interface entre o PRQ e o FDF utilizada para a submissão de pedidos e produtos orbitais pelo FDF e envio de respostas pelo PRQ.
- *iPDS* – interface entre o PRQ e a PDS, utilizada pelo PRQ para aceder à PDS tanto para armazenar os ficheiros recebidos e gerados como para acedê-los para leitura ou edição.

Para as interfaces partilhadas pelo PRH e pelo PRU recorreu-se ao padrão de desenho Singleton. Deste modo garantimos que ambos os componentes acedem a um único ponto de forma global e consistente.

O padrão Singleton é usado para restringir a instanciação de uma classe a um único objecto. É essencialmente útil quando exactamente um único objecto é necessário para coordenar acções dentro de um sistema. Uma das formas de implementar o padrão Singleton é através da criação de uma classe com um método que cria uma nova instância da classe se não existir já outra. Se uma instância já existir, simplesmente retorna uma referência para essa mesma instância. O construtor da classe deverá ter acesso *private* ou *protected* de modo a garantir que não pode ser instanciado por outra classe.

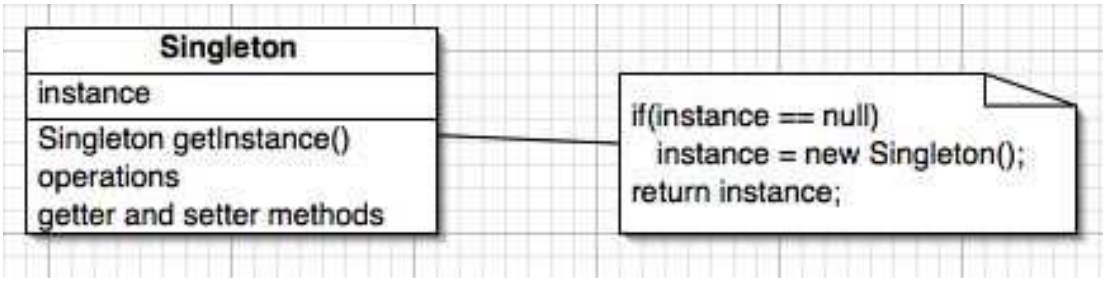

Figura 5: Padrão de desenho Singleton

3.2.1.1. Análise sintática de XML

As funções do *Planning Request Handler* centram-se na manipulação de ficheiros XML. Para suportar essas funções recorreu-se a uma biblioteca de análise e validação de XML, o *Xerces*, nomeadamente a implementação em Java.

O *Xerces* consiste numa biblioteca que disponibiliza analisadores sintáticos de XML. Esta biblioteca implementa um número de APIs padrões para a análise sintática de XML, incluindo DOM, SAX e SAX2. Foi importante compreender o modo de funcionamento do DOM e do SAX para perceber quais as diferenças e vantagens de um em relação ao outro.

O seguinte documento XML, books.xml, descreve um catálogo de livros e é usado nos próximos exemplos:

```
<catalog>
<!—Sample —>
  <book id="101">
    <title>XML in a Nutshell</title>
    <author>Elliotte Rusty Harold, W. Scott Means</author>

    <price>39.95</price>
  </book>
  <book id="121">
    <title>Who Moved My Cheese</title>
    <author>Spencer, M.D. Johnson, Kenneth H. Blanchard</author>
    <price>19.95</price>
  </book>
</catalog>
```

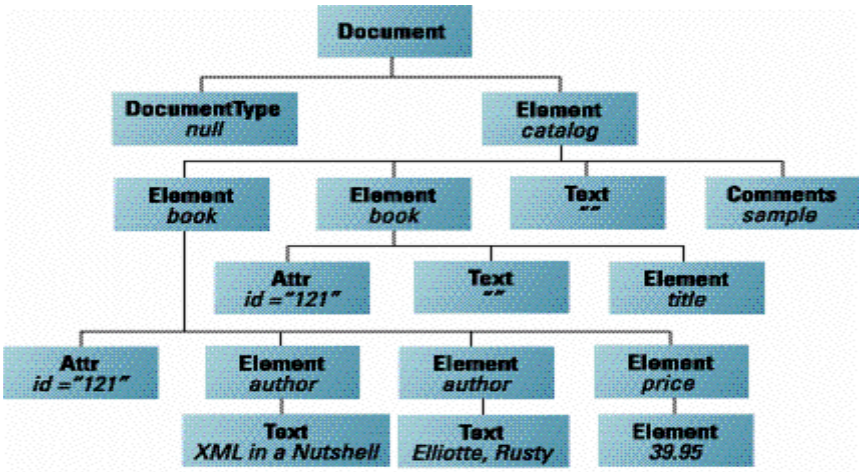

Figura 6: Árvore contruída por um analisador sintático DOM

O *Document Object Model (DOM)* descreve um documento XML numa estrutura em árvore, constituindo cada elemento XML um nó da árvore. Um parser DOM lê todo o documento e cria a respectiva árvore em memória. Esta árvore é formada por classes que implementam a interface `org.w3c.dom.Node`. Esta interface fornece métodos de acesso e modificação dos nós da árvore.

Esta abordagem do DOM tem a vantagem de facilitar a manipulação dos valores lidos. Contudo, os custos implicados a nível de memória não o torna uma hipótese viável para o *Planning Request Handler*.

O SAX, *Simple API for XML*, é um analisador sintático tradicional cujo comportamento é modificado através de eventos. O SAX lê o documento XML de modo incremental, efectuando chamadas às funções da aplicação sempre que reconhece um sinal. Os eventos são lançados no início e no fim de um documento, no início e no fim de um elemento, e por aí adiante. Os métodos que gerem estes eventos estão definidos na interface `org.xml.sax.ContentHandler`, que deverá ser implementada em qualquer aplicação que faça uso do SAX.

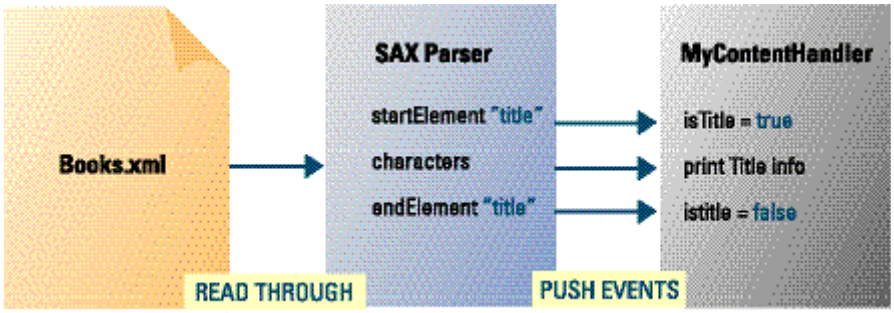

Figura 7: O analisador SAX: leitura e eventos

Como podemos ver na Figura 7, o SAX lança eventos no início e no fim do elemento "title" que são geridos pelos métodos startElement e endElement, respectivamente. Os métodos que gerem os eventos lançados pelo SAX são implementados no MyContentHandler (que implementa a classe já referida ContentHandler). O método *characters* gere a leitura de uma cadeia de carateres, isto é, devolve um vector com os caracteres lidos entre o início e fim de um elemento; para o elemento "title", a cadeia de caracteres devolvida seria "XML in a Nutshell".

O *Planning Request Handler* utiliza um analisador sintático SAX para ler e escrever ficheiros XML, o que implica o conhecimento prévio da sua estrutura.

Para compreender melhor o modo como funcionava o *Xerces*, nomeadamente o SAX, foram realizados pequenos programas que usavam ficheiros XML com exemplos simples.

O conhecimento do modo de funcionamento da análise sintática do SAX foi bastante importante para a posterior fase de implementação, visto que grande parte do funcionamento do PRQ centra-se na manipulação de XML. A manipulação de XML no PRQ engloba a leitura e escrita de ficheiros, ou de parte destes.

3.2.1.2. Validação XML

A validação de um documento XML permite verificar se o mesmo segue uma estrutura definida num *schema* XML.

Um *schema* XML é uma representação, baseada em XML, da estrutura de um documento XML. Através do seu suporte para tipos de dados e *namespaces*, o *schema* XML tem o potencial de fornecer a estrutura padrão para elementos XML e seus atributos. Para verificar que um documento segue a estrutura definida num *schema* XML é necessário que seja validado com esse mesmo *schema*. O documento XML a validar é tecnicamente chamado de instância de um documento.

A validação é efectuada usando um analisador sintático. No caso do PRQ foi usado o SAX. Para validar o documento XML é necessário saber a localização do *schema* XML, que deverá estar declarada no elemento raíz da instância através do atributo `xsi:schemaLocation` ou `xsi:noNamespaceSchemaLocation`, dependendo se o *schema* tem *namespaces* ou não. A validação do documento é efectuada paralelamente à análise sintática. Para tal é somente necessário activar o modo de Validação do analisador de modo a serem reportados os erros de validação.

Como já foi referido o servidor do PRQ tem como principal função validar os ficheiros que recebe, de modo a que apenas os pedidos válidos possam entrar no sistema de planeamento. Nesta fase foi importante compreender o modo como a validação era processada, ou melhor, inicializada, e qual o papel do *schema* neste processo. Chegou-se à conclusão que os analisadores sintáticos implementados pelo Xerces têm um conjunto de funcionalidades associadas que podem ser activadas entre as quais se encontram a validação. Assim para validar um documento é apenas necessário criar uma instância do analisador sintático, o SAX neste caso, e activar a validação através

3.2.2 Planning Request Handler User Interface (PRU)

O cliente do Planning Request Handler permite a submissão de pedidos por parte do operator do SCPF, assim como a edição de pedidos armazenados na *Planning Data Store*. O PRU estará integrado numa aplicação gráfica mãe, a do SCPF, que servirá de suporte a todas as fases do planeamento da missão.

Na fase de desenho foi concebido um protótipo da GUI do PRQ (Figura 8). Esta fase permite identificar a nível gráfico quais os campos indispensáveis, assim como a sua localização e disposição, que permitirão ao utilizador final utilizá-la sem dificuldades. Foram seguidas as *guidelines* para as interfaces gráficas definidas no âmbito do Galileo. A maioria das *guidelines* aplicáveis fazem parte também do conjunto de requisitos do PRQ.

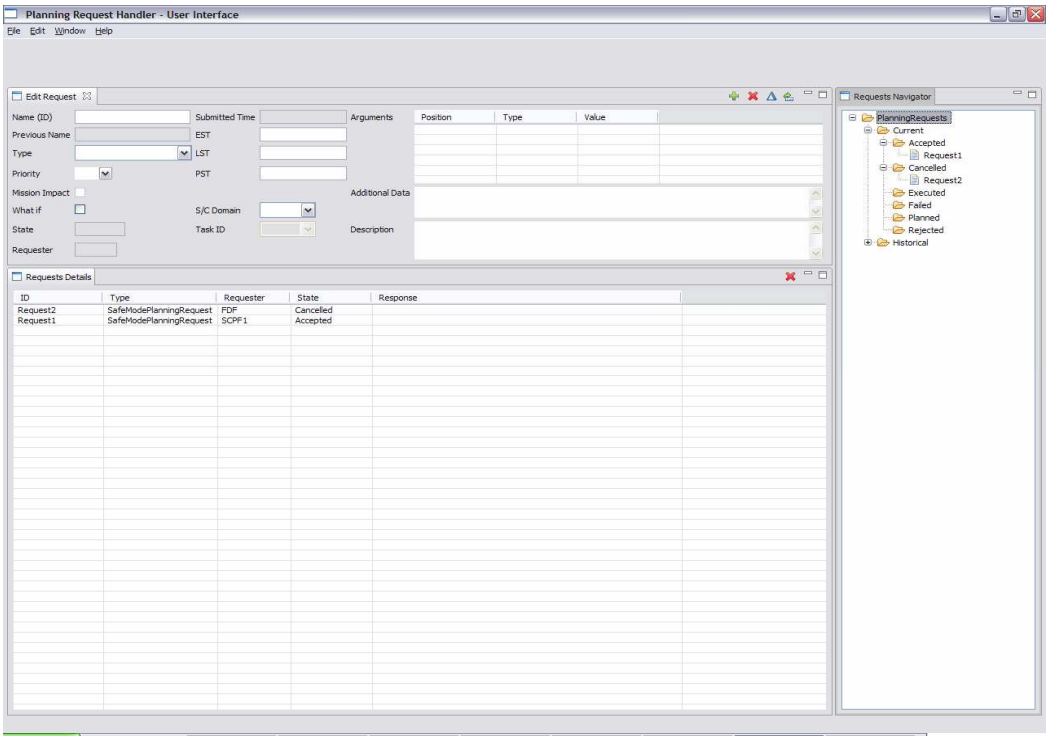

Figura 8: Protótipo da GUI do PRQ

Os protótipos foram desenvolvidos em Java com a tecnologia Eclipse Rich-Client Plataform (Eclipse RCP), e foram apresentadas no documento de desenho detalhado.

As classes que faziam parte do protótipo do PRU foram incluidas no modelo UML através do processo de *reverse engineering*.

3.2.2.1. Eclipse RCP

A plataform RCP do Eclipse fornece uma forma simples, poderosa e flexível de desenvolver aplicações gráficas utilizando a linguagem de programação Java. Esta tecnologia define os seguintes conceitos básicos:

- *Workbench* – termo utilizado para a interface gráfica genérica do Eclipse. O *workbench* consiste na janela gráfica à qual são adicionadas perspectivas
- *Perspectiva* – conjunto de *views*, editores e menus incluindo as suas posições e tamanhos. O PRQ tem apenas uma perspectiva.
- *View* – *views* são partes amovíveis que constituem a maioria do conteúdo do *workbench*. Na Figura 8 vemos duas *views*, uma na base da janela, a tabela dos pedidos e outra à direita, *view* de navegação de pedidos.
- *Editores* – partes que têm um input associado dentro de um *workbench*. O editor de pedidos do PRU permite visualizar e editar pedidos.

O Eclipse RCP é a funcionalidade que se encontra por trás do *core* da plataforma Eclipse. Se separarmos as partes do Eclipse que o tornam num IDE, ficamos com um conjunto que fornece as funcionalidades principais ao *workbench* incluindo o suporte para editores e *views*, menus, barras de ferramentas, tabelas, árvores, etc.

O Eclipse RCP fornece:

- Um *look and feel* nativo e consistente para as aplicações e *features*;

- Serviços comuns a todas as aplicações como gestão de janelas, ajuda;
- Extensibilidade.

O Eclipse RCP utiliza o toolkit gráfico SWT (*Standard Widget Toolkit*) no lugar do tradicional Swing. O SWT utiliza os controlos e diálogos do sistema operativo sobre o qual a Java Virtual Machine (JVM) está a ser executada.

Além do SWT, o Eclipse usa a biblioteca JFace, que estende o SWT básico, adicionando-lhe novos diálogos, controlos e classes úteis.

Tanto o SWT como o JFace podem ser utilizados fora da plataforma Eclipse.

A Figura 9 mostra a relação entre os componentes da plataforma RCP.

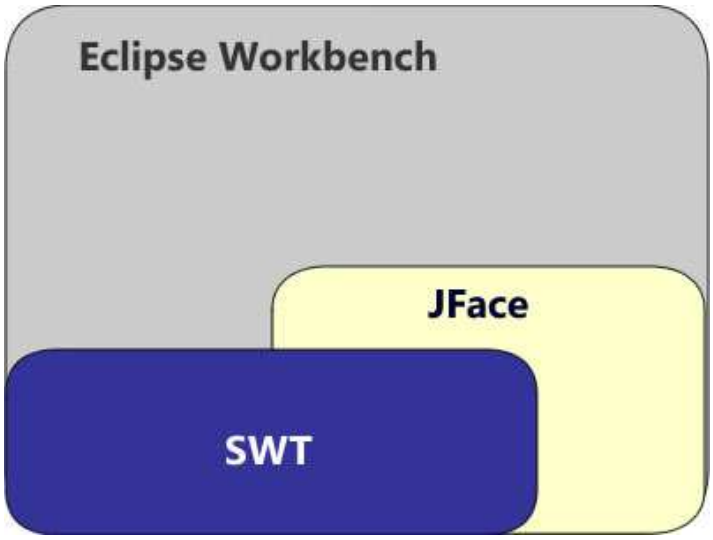

Figura 9: Eclipse Workbench

3.2.3 Ferramentas Utilizadas

Para a criação do modelo detalhado foi utilizada a ferramenta *Enterprise Architect*. O *reverse enginnering* das classes do protótipo do PRU foi efectuado utilizando, também, o *Enterprise Architect*. O *reverse enginnering* consistiu, neste caso, na conversão de código em Java para diagramas UML.

O protótipo do PRU foi desenvolvido utilizando o Eclipse, com a tecnologia Eclipse RCP e a versão 1.5 do Java.

O documento do desenho detalhado do PRQ foi produzido com a ferramenta Word do MS Office.

3.3. Implementação

O resultado da fase de implementação deverá satisfazer os requisitos do modo especificado no desenho detalhado, promovendo a sua manutenção através de padrões de programação.

A implementação do *Planning Request Handler* seguiu de modo geral a arquitectura definida na fase de desenho, essencialmente a nível de organização de componentes. Contudo sentiu-se a necessidade de efectuar algumas modificações a nível de classes, principalmente na criação de novas classes que suportassem funcionalidades não descritas no modelo inicial.

Foram seguidas algumas convenções e regras definidas no GSWS na codificação, como nome e inicialização das variáveis, e mesmo formatação do código.

Devido à existência de indefinições a nível de interfaces, e à dependência de componentes externos ainda não disponibilizados, assim como a atrasos sofridos no projecto, alguns dos requisitos não foram implementados. Os requisitos não implementados estão documentados no relatório dos testes de aceitação.

O desenho detalhado e requisitos serviram como *input* a esta fase.

O *output* desta fase consistiu no código produzido.

3.3.1 Implementação do Planning Request Handler User Interface

O trabalho a nível da implementação da interface gráfica do PRQ foi desenvolvido a partir do protótipo realizado na fase de desenho.

Ao protótipo inicial foi acrescentado todo o processamento necessário ao funcionamento do cliente: acesso aos dados armazenados para visualização, criação de pedidos e submissão dos mesmos ao servidor, visualização de respostas, edição e cancelamento de pedidos já aceites.

Nesta fase foi adquirido um maior conhecimento da tecnologia Eclipse RCP, essencialmente a nível de manipulação dos dados a serem apresentados ao operador.

3.3.2 Implementação do Planning Request Handler Server

A parte do modelo obtido na fase de Desenho Detalhado referente a este componente foi na sua maioria seguido. À medida que a implementação evoluía, o modelo sofreu algumas alterações de modo a colmatar falhas que existiam.

Nesta fase foi adquirido um maior conhecimento das funcionalidades fornecidas pela biblioteca *Xerces*.

3.3.3 Ferramentas utilizadas

O PRQ foi desenvolvido em Java, versão 1.5, utilizando o Eclipse como IDE.

Uma vez que a maioria das funcionalidades requeriam a manipulação de ficheiros XML, foram gerados alguns exemplos a partir dos schemas existentes. Ferramentas como o Altova XML Spy e o Stylus Studio foram utilizadas para gerar ficheiros XML.

3.4. Generic Planning Request Service (GPRS)

Durante o estágio, e após a fase de implementação do *Planning Request Handler* foi produzida uma ferramenta de teste chamada *Generic Planning Request Service*. Esta ferramenta tinha como principais funcionalidades a geração de ficheiros de XML, pedidos de planeamento e *mid-term plans*, recebidos pelo PRQ, e a sua submissão ao servidor. O principal propósito desta ferramenta era submeter no servidor pedidos de planeamento e *mid-term plans* de forma a testar o seu funcionamento.

Perante a inexistência de qualquer documento de requisitos desta ferramenta, foi discutido com o cliente quais as suas funcionalidades e quais as possibilidades de reutilizar algum do trabalho já realizado a nível de implementação. Aqui foram identificados alguns paralelismos com o componente PRU: a criação de ficheiros XML, no caso do PRU restrito à criação de pedidos de planeamento, e a submissão de ficheiros ao servidor. Assim, concluiu-se que o trabalho a realizar consistiria na extensão das

funcionalidades do PRU, nomeadamente à adição da funcionalidade de criação de *mid-term plans*. Visto que os *mid-term plans* são formados por pedidos de planeamento e informação acerca da disponibilidade das estações TT&C, já cobriamos parte da sua criação, a referente aos pedidos. Optou-se então por criar um novo editor para a informação da disponibilidade das estações. Desta forma são gerados ficheiros independentes para os pedidos e disponibilidades das estações.

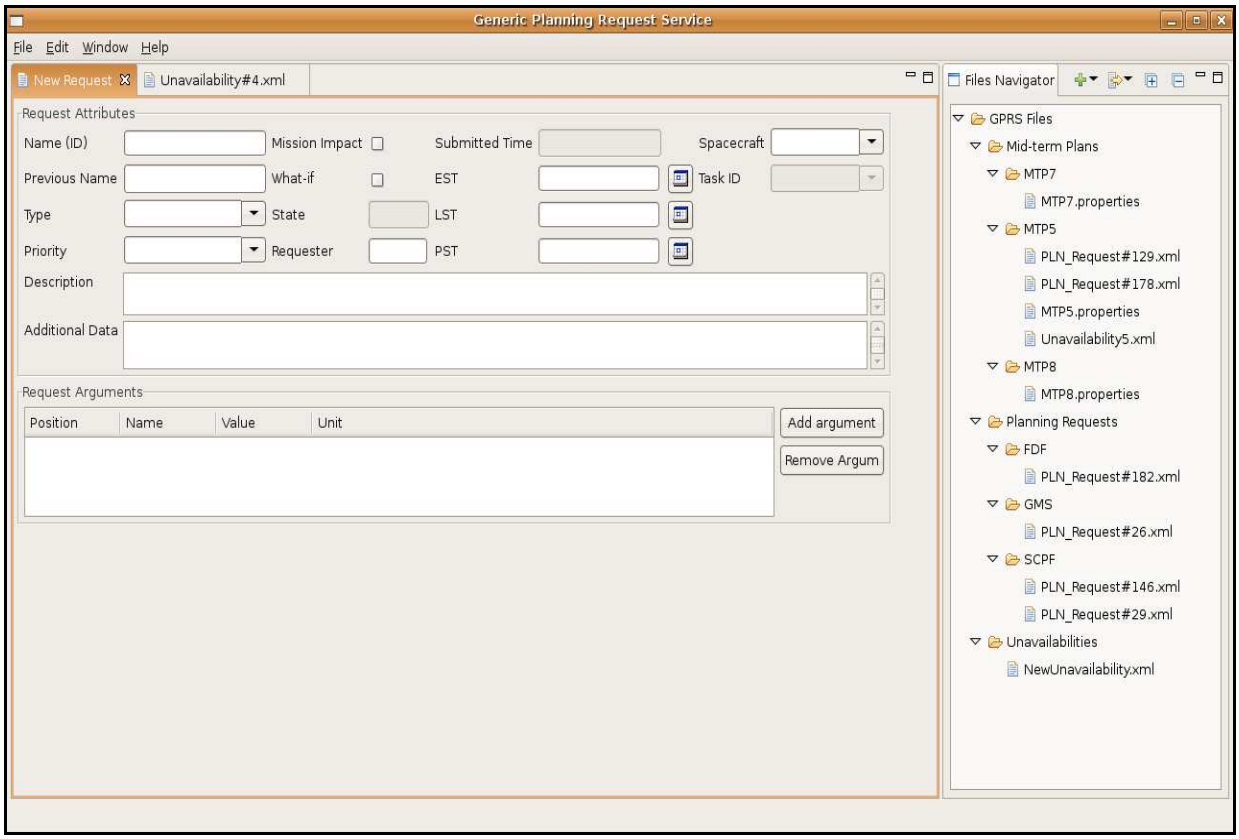

Figura 10: Screenshot do GPRS

Os ficheiros criados são organizados pelo seu tipo. Na Figura 10 podemos visualizar a árvore de navegação do lado direito, que apresenta os ficheiros criados. Ao criar um *mid-term plan* é criada uma nova directoria na árvore de navegação sob o nó pai correspondente. Os ficheiros sob a nova directoria, que poderão ser pedidos de planeamento e informação da disponibilidade das estações, serão incluidos aquando da geração do ficheiro do *mid-term plan*. Para adicioná-los ao MTP é apenas necessários arrastar o ficheiro na árvore para dentro da directoria do MTP.

Os dados introduzidos pelo operador não são validados, pois a ferramenta deverá permitir tanto a introdução de dados válidos como inválidos para efeito de teste.

Esta ferramenta constitui um bom exemplo de reutilização de código, resultando na economização de esforço na sua implementação.

3.4.1 Ferramentas utilizadas

Ver 3.3.3.

3.5. Verificação e Validação

O processo de verificação e validação tem o propósito de verificar se o sistema está a ser desenvolvido da forma correcta e se realiza as funções supostas.

Este processo foi, no fundo, aplicado durante toda a fase de desenvolvimento. À medida que novas funcionalidades, especificadas anteriormente, eram implementadas, era feita a validação da mesma verificando que funcionava do modo suposto. A execução de testes é o modo formal de fazer verificação e validação.

Na fase posterior ao desenho foram criados os planos referentes aos testes de aceitação e testes unitários.

3.5.1 Planos de teste

Os planos de teste pretendem fornecer suporte à execução dos testes, tanto quanto ao modo como são executados como à identificação do seu sucesso ou falha. Os planos de testes de aceitação e de testes unitários foram elaborados após a fase de desenho detalhado e antes da fase de implementação.

3.5.1.1. Unit Test Plan

Os testes unitários visam testar pequenas partes ou unidades do sistema. O principal objectivo é o de encontrar falhas de funcionamento dentro de uma pequena parte do sistema, funcionando este independentemente do todo. Esta aproximação permite uma maior facilidade na detecção de *bugs*, visto que existe um maior isolamento do módulo a testar. A implementação dos testes unitários deverá ser realizada à medida que o produto é implementado, conduzindo a uma detecção antecipada das falhas. Um dos aspectos mais importantes dos testes unitários é o de garantirem a correcta implementação do método caso este seja alterado, pois funcionam como uma especificação do sistema. Se temos um teste que diz $a+b=c$, então sem dúvida que aquele será o comportamento do sistema. É uma forma que retrata as especificações da forma mais concreta possível. Outro factor importante dos testes é a análise por valor fronteira que permite verificar que os valores aceites pela unidade estão no seu domínio, rejeitando os restantes; esta análise é implementada através de testes que põem à prova os valores fronteiriços.

As unidades consideradas no plano de testes do *Planning Request Handler* foram os métodos das classes a ser implementadas. Para cada método, foram especificados um ou mais testes, de modo a permitir testar se o mesmo se comporta correctamente de acordo com o input que lhe é dado. Para cada teste unitário do *Planning Request Handler* foram descritos os campos definidos na seguinte tabela, excepto o último ("Requisitos a serem verificados"):

Campo	Definição
Identificador	Identificador único atribuído ao caso de teste.
Nome	Nome atribuído ao caso de teste. Deve ser sugestivo de modo a identificar facilmente o propósito do teste.
Especificação	Descreve o teste e os passos para a sua execução.
Critérios de passagem	Define o que valida a passagem de um teste, isto é, para um teste ser bem sucedido deve verificar os critérios de passagem.
Ferramentas	Ferramentas utilizadas para a execução do teste. Exemplo:

Campo	Definição
	Junit.
Dados de teste	Dados de input (ficheiros, cadeias de caracteres, objectos) para o método a ser testado.
Entrega	Fase de entrega do sistema sobre qual já deverá ter sido executado o teste em questão.
Fase de teste	Define em que fase do desenvolvimento do componente este teste deverá ser executado. Por exemplo a nível de componente, quando este é desenvolvido ou a nível de sistema, quando o componente já está integrado com outros módulos do sistema.
Requisitos a serem verificados	Requisitos verificados por este caso de teste.

Tabela 2: Campos incluídos nos planos de teste

O plano de testes unitários sofreu algumas alterações, no sentido de acrescento, aquando das modificações efectuadas no desenho já na fase de implementação. O objectivo era manter todos os documentos coerentes, de modo a que esta fase de verificação e validação não sofresse atrasos.

Junit foi a ferramenta utilizada para a implementação dos testes unitários. Esta constitui uma framework em Java usada para escrever e correr testes automaticamente.

A realização do plano de testes unitários teve como *input* o desenho detalhado, no qual estavam especificadas as unidades que compunham o sistema.

O output constitui o *Unit Test Plan*, que consiste, neste caso, em uma folha Excel com o conjunto de caso de testes a executar.

3.5.1.2. Acceptance Test Plan

O *Acceptance Test Plan* descreve um conjunto de testes que deverão ser efectuados pelo cliente de modo a que este assegure que todos os requisitos do sistema são satisfeitos, e por conseguinte, determinar se o sistema é aceite. No fundo, trata-se de aceitar ou rejeitar um produto final.

O plano de testes apresenta uma descrição detalhada de cada caso de teste que permite a fácil compreensão da equipa técnica que irá pôr em prática o plano.

A informação descrita no *Acceptance Test Plan* relativa a cada caso de teste é análoga à do *Unit Test Plan*: ver Tabela 2 , incluindo o campo "Requisitos a serem verificados".

Ao fazer os plano de testes, identificaram-se alguns que podiam ser automatizados, numa abordagem semelhante aos testes unitários, utilizando a ferramenta Junit.

Para a formulação do plano de testes de aceitação o input foi o conjunto de requisitos do *Planning Request Handler*.

O output constitui o *Acceptance Test Plan*, que consiste, neste caso, em uma folha Excel com o conjunto de caso de testes a executar após a implementação.

3.5.2 Execução dos testes

A execução dos testes unitários e de aceitação foram efectuados num ambiente, tanto a nível de Hardware como a nivel de Software, bastante aproximado do que realmente virá a ser utilizado. Esta fase foi realizada tendo como suporte os planos de testes definidos.

Os testes deverão ser executados de acordo com a sua especificação, sendo a sua passagem ou falha validada pela satisfação dos critérios de passagem.

3.5.2.1. Unit tests

Os testes unitários foram criados utilizando a ferramenta Junit.

O Junit é uma framework open-source que possibilita a criação de testes unitários em Java. Muitos IDEs, como o Eclipse incorporam o Junit dentro do seu ambiente de desenvolvimento, facilitando o uso dessa framework. De qualquer modo é possível correr testes e recolher os seus resultados numa consola.

O Junit permite criar classes de teste. Estas classes contêm um ou mais métodos para que sejam realizados os testes, os quais são chamados automaticamente. Esta forma automatizada de executar os testes permite construir uma bateria de testes que possam ser executados em qualquer fase da implementação de modo a verificar que o sistema está a ser correctamente construído.

O Eclipse permite visualizar graficamente, de uma forma bastante fácil e rápida, se um teste passou ou falhou. Nos casos de falha é assinalado qual o método de teste que falhou e qual o tipo de falha. Esta apresentação gráfica permitiu facilitar a leitura dos resultados dos testes aquando da sua execução.

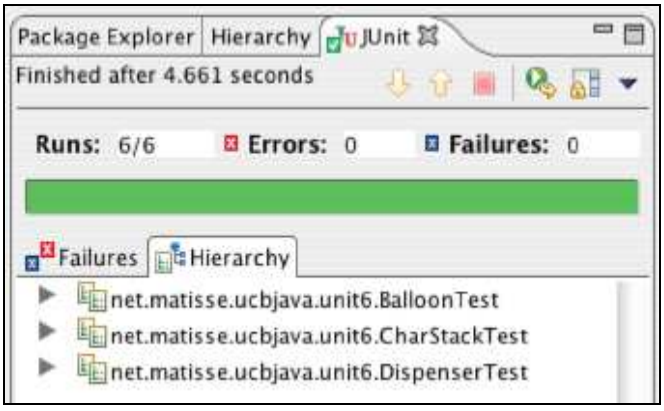

Figura 11: Exemplo de um teste bem sucedido.

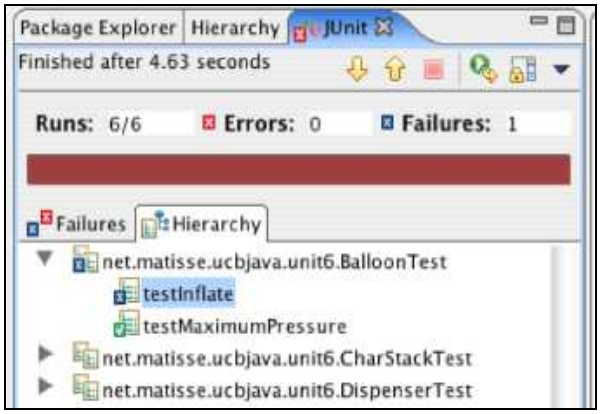

Figura 12: Exemplo de um teste falhado

3.5.2.2. Acceptance tests

Os testes de aceitação têm uma abordagem um pouco diferente dos testes unitários. São executados quando o sistema já está implementado e têm o objectivo de verificar que todos os requisitos foram implementados.

Alguns dos testes de aceitação que foram automatizados foram implementados de forma semelhante aos testes unitários.

A maioria dos testes de aceitação são executados de forma manual, como por exemplo nos testes de funcionalidades do cliente e do servidor do PRQ. Outros são realizados através de inspecções ao código ou da interface gráfica do PRU.

3.5.3 Test Reports

Os resultados dos testes foram documentados num relatório próprio. O relatório de testes deve também descrever o ambiente em que esses foram executados.

Para cada caso de teste foram documentados os seguintes campos:

Campo	Definição
Identificador	Identificador do correpondente caso de teste.
Nome	Nome do caso de teste executado.
Successo/Falha	Estado obtido da execução do teste. O sucesso/falha de um teste resulta da satisfação dos critérios de passagem.
Problemas encontrados	Este campo poderá ser utilizado para descrever as razões de falha de um teste, ou problemas encontrados na sua execução.

Como já foi referido, alguns testes de aceitação não foram executados. Nestes casos o campo "Problemas encontrados" descreve a razão da não execução do teste respectivo.

3.5.4 Ferramentas utilizadas

As ferramentas utilizadas nesta fase foram o Eclipse com o Junit integrado.

O Excel e o Word do MS Office foram utilizados para os documentos de planos de testes e relatórios de teste, respectivamente.

3.6. Elaboração do Manual do Utilizador

Houve uma contribuição na elaboração do Manual do Utilizador, que no seu todo ficou à responsabilidade do cliente. A contribuição consistiu essencialmente na produção da secção referente ao *Planning Request Handler User Interface*, descrevendo as suas principais operações e os passos a serem executados.

Capítulo 4

Conclusão

Este capítulo constitui uma apreciação e conclusão do trabalho desenvolvido e do estágio.

4.1. Apreciação crítica do trabalho desenvolvido

Como já foi referido ao longo deste documento, o projecto SCPF sofreu alguns atrasos ao longo do estágio, que afectaram o planeamento original e o resultado final do estágio.

No início do trabalho estavam definidas duas entregas do *Planning Request Handler*, a D1 e a D2. Contudo, alguns desenvolvimentos no planeamento definiram que o PRQ deveria ser construído numa única fase, sendo entregue na D1. No final do estágio a D1 ainda não tinha sido realizada, estando ainda longe de poder ser efectuada.

Os atrasos no projecto, completamente alheios à Critical Software, deveram-se essencialmente à instabilidade sentida na definição de uma *baseline* coerente. A *baseline* comporta todos os documentos necessários na elaboração dos projectos, nos quais estão definidos os protocolos a utilizar, o formato e conteúdo das mensagens a trocar entre componentes, etc.

De forma a evitar que os atrasos tivessem um grande impacto no estágio foi assumido que o trabalho iria ser desenvolvido, nomeadamente a fase de implementação, de acordo com a documentação existente, implementando todas as funcionalidades possíveis. Daqui resultou a elaboração de um novo plano de estágio.

Durante a fase de implementação o trabalho sofreu alguns atrasos, nomeadamente devido à falta de experiência em algumas tecnologias e à deficiência na análise do tempo necessário para a realização das tarefas.

O resultado final do trabalho desenvolvido foi bastante razoável, a maioria dos requisitos foram implementados e verificados, tendo-se obtido um produto funcional, que poderá de certo ser aproveitado quando o arranque oficial da fase de implementação tomar lugar.

4.2. Trabalho futuro

Devido à instabilidade da *baseline* e às discussões que ainda estão a ter lugar, esperam-se ainda mais atrasos no projecto. Os pedidos mais recentes de alterações na baseline referiam mudanças com grande impacto no *Planning Request Handler* que iriam aumentar a sua complexidade, resultando num aumento de esforço aplicado.

Num futuro imediato, existem alguns requisitos a implementar, como as funcionalidades de *logging* e *trace* e algumas melhorias a serem consideradas.

No final do estágio foram documentadas quais as modificações importantes e obrigatórias a realizar sobre o PRQ, assim como algumas sugestões e possíveis modificações a efectuar.

4.3. Apreciação do estágio

O resultado final do estágio é considerado bastante positivo. O ambiente na empresa é muito bom e os meus colegas mostraram-se sempre prontos a ajudar e receptivos ao esclarecimento das dúvidas. Além da vertente profissional, desenvolveu-se uma relação pessoal de companheirismo, e nalguns casos, até de amizade. As condições de trabalho foram sempre de encontro às minhas necessidades desde o primeiro dia do estágio.

O estágio revelou-se uma experiência de aprendizagem bastante rica. A inserção num projecto com um plano estabelecido, com o qual contactei nas diversas fases de desenvolvimento, deu-me uma visão muito mais realista do que é produzido em cada uma das fases e quais os métodos e metodologias utilizados. As reuniões semanais de projecto, as reuniões com os clientes, as revisões dos documentos, as auditorias ao projecto constituíram eventos muito importantes fornecendo-me noções de gestão tanto a nível de projecto como de qualidade, assim como a aquisição de competências nestas áreas.

O facto da documentação ser toda escrita em inglês, e de a comunicação com o cliente ser efectuado em inglês permitiu melhorar o meu nível de expressão, essencialmente escrita, nessa língua.

Durante o estágio foi realizada uma *workshop* intitulada *"Quality for Newcomers"*, que reflecte a aposta da CSW na área da qualidade promovendo o conhecimento dos processos da empresa entre os novos trabalhadores e estagiários. Nesta workshop foram abordadas as várias fases de um projecto, enfatizando os métodos adoptados que garantem a obtenção de um bom produto final.

Os conhecimentos adquiridos a nível académico foram bastante úteis essencialmente no que toca à capacidade de investigação e aprendizagem em temas desconhecidos ou pouco dominados. Muitos dos conhecimentos adquiridos na Licenciatura em Engenharia Informática revelaram-se importantes para o estágio, principalmente a nível de programação na linguagem Java. Também os conceitos a nível de desenho, de gestão de qualidade, de verificação e validação foram relevantes.

Contudo, considero que o curso peca em vertentes muito importantes no desenvolvimento de software: análise de requisitos, desenho detalhado, e validação e verificação. A maioria dos projectos desenvolvidos durante o curso centram-se no produto fruto da implementação, quando é geralmente conhecido que são as fases preliminares as mais importantes para garantirem um produto de qualidade. Também considero que deveria ser dada uma maior importância às fases de validação e verificação, promovendo a boa prática de serem executados testes à medida que são acrescentadas funcionalidades durante a implementação.

Bibliografia

[1] Enterprise Architect:
http://www.sparxsystems.com.au/products/ea.html

[2] Documentação da plataforma Java 1.5: http://java.sun.com/j2se/1.5.0/docs/

[3] Eclipse IDE: http://www.eclipse.org/

[4] Eclipse RCP: http://www.eclipse.org/home/categories/rcp.php,
Lemieux, J., McAffer, J., *Eclipse Rich Client Platform: Designing, Coding, and Packaging Java(TM) Applications*,
Addison-Wesley, 2005.

[5] Junit: http://www.junit.org/index.htm

[6] Xerces Java Parser: http://xerces.apache.org/xerces-j/

[7] CVS: http://pt.wikipedia.org/wiki/CVS

[8] Johnson M., JavaWorld, *Programming XML in Java*:
http://www.javaworld.com/jw-03-2000/jw-03-xmlsax.html

[9] Braude, E. J., *Software Engineering: An Object-Oriented Perspective*, John Wiley & Sons, Inc.

[10] Guo P., Basu, J., Scardina, M., and Karun K., Oracle, *Parsing XML Efficiently*:
http://www.oracle.com/technology/oramag/oracle/03-sep/o53devxml.html

Índice Remissivo